

# DEEP DENOISING AUTO-ENCODER FOR STATISTICAL SPEECH SYNTHESIS

Zhenzhou Wu\*

School of Computer Science  
McGill University, Canada

Shinji Takaki, Junichi Yamagishi

National Institute of Informatics  
Japan

## ABSTRACT

This paper proposes a deep denoising auto-encoder technique to extract better acoustic features for speech synthesis. The technique allows us to automatically extract low-dimensional features from high dimensional spectral features in a non-linear, data-driven, unsupervised way. We compared the new stochastic feature extractor with conventional mel-cepstral analysis in analysis-by-synthesis and text-to-speech experiments. Our results confirm that the proposed method increases the quality of synthetic speech in both experiments.

**Index Terms**— Speech synthesis, HMM, DNN, Auto-encoder

## 1. INTRODUCTION

Current statistical parametric speech synthesis typically uses hidden Markov models (HMMs) to represent probability densities of speech trajectories given text [1]. This is a well-established method and it is straightforward to apply this framework for new languages. It also offers interesting advantages in terms of flexibility and compact footprint [2, 3, 4, 5]. It is known, however, that speech synthesized from statistical models still sounds somehow artificial and less natural compared to speech synthesized by the best unit selection systems.

It is often said that averaging in statistical synthesis systems partly removes spectral fine structure of natural speech, and thus there is room for the improving the segmental quality. A stochastic postfilter approach [6] proposes to use a deep neural network (DNN) to model the conditional probability of the spectral differences between natural and synthetic speech. The approach is able to reconstruct the spectral fine structure lost during modeling and has achieved significantly quality improvement for synthetic speech [6]. In this experiment, the HMM-based speech synthesiser was trained in the mel-cepstral domain, while the DNN-based postfilter was trained in the spectral domain.

This indicates that the current statistical parametric speech synthesis suffers from quality loss due to statistical averaging in the mel-cepstral domain, but also due to conversion from high-dimensional spectral features to lower dimensional mel-cepstral parameters and hence this brings us a new question: are current intermediate representations such as mel-cepstral coefficients appropriate for statistical training of acoustic models? Can we automatically find a more appropriate intermediate representation that suits acoustic modelling and results in better quality of synthetic speech?

---

The first author performed this work while at the National Institute of Informatics, Japan.

This work was supported in part by EPSRC through Programme Grant EP/I031022/1 (NST) and EP/J002526/1 (CAF). Shinji Takaki was supported in part by NAVER Labs.

To answer this question, this paper proposes a DNN-based feature extraction method. More specifically we propose to use a deep denoising auto-encoder technique as a non-linear robust feature extractor for speech synthesis and apply it to high-dimensional spectral features obtained from STRAIGHT vocoder [7]. We compare this data-driven, unsupervised feature extraction approach with the conventional mel-cepstral analysis, which is based on a linear discrete cosine transform of the log spectrum.

This paper is organised as follows: in Section 2, we outline related DNN-based approaches and in Section 3 we describe the proposed deep denoising auto-encoder technique. In Section 4, we mention how we train the model and the experimental conditions and evaluation results are shown in Section 5. Discussions and the summary of our findings are given in Section 6.

## 2. RELATED WORK USING DNN AND AUTO-ENCODER

This section overviews related work using DNN and/or auto-encoder in the speech information processing field. DNN has been applied for acoustic modelling of speech synthesis. For instance [8] uses DNN to learn the relationship between input texts and the extract features instead of decision tree-based state tying. Restricted Boltzmann machines or deep belief networks have been used for modelling output probabilities of HMM states instead of GMMs [9]. Recurrent neural network or long-short term memory was used for prosody modelling [10] or acoustic trajectory modelling [11].

To the best of our knowledge, this is the first work to use deep denoising auto-encoder for speech synthesis, but, deep auto-encoder based bottleneck features are used by several groups for ASR [12, 13] and deep denoising auto-encoder is also verified for noise-robust ASR [14] or reverberant ASR tasks [15, 16].

Techniques that are closely related to this paper are a spectral binary coding approach using deep auto-encoder proposed by Deng et al [17] and a speech enhancement approach using deep denoising auto-encoder where they try to reconstruct clean spectrum from noisy spectrum [18]. The approach proposed here is also related to heteroscedastic linear discriminant analysis (HLDA) [19, 20] and probabilistic linear discriminant analysis (PLDA) [21, 22, 23]. Our key idea is however different from these as we use deep auto-encoder based continuous bottleneck features calculated from spectrum to reconstruct high-quality synthetic speech.

## 3. AUTO-ENCODER

### 3.1. Basic Auto-encoder

Auto-encoder is an artificial neural network that is used generally for learning a compressed and distributed representation of a dataset. It consists of the encoder and the decoder. The encoder maps a input

vector  $\mathbf{x}$  to a hidden representation  $\mathbf{y}$  as follows:

$$\mathbf{y} = f_{\theta}(\mathbf{x}) = s(\mathbf{W}\mathbf{x} + \mathbf{b}), \quad (1)$$

where  $\theta = \{\mathbf{W}, \mathbf{b}\}$ .  $\mathbf{W}$  and  $\mathbf{b}$  represent a  $m \times n$  weight matrix and a bias vector of dimensionality  $m$  respectively, where  $n$  is the dimension of  $\mathbf{x}$ . The function  $s$  is a non-linear transformation on the linear mapping  $\mathbf{W}\mathbf{x} + \mathbf{b}$ . Frequently  $s$  is a sigmoid, a tanh, and a relu function.  $\mathbf{y}$ , the output of the encoder, is then mapped to  $\mathbf{z}$ , the output of the decoder. The mapping is performed by a linear function alone that employs a  $n \times m$  weight matrix  $\mathbf{W}'$  and a bias vector of dimensionality  $n$  as follows:

$$\mathbf{z} = g_{\theta'}(\mathbf{y}) = \mathbf{W}'\mathbf{y} + \mathbf{b}', \quad (2)$$

or a linear mapping followed by a non-linear transformation  $t$ ,

$$\mathbf{z} = g_{\theta'}(\mathbf{y}) = t(\mathbf{W}'\mathbf{y} + \mathbf{b}'), \quad (3)$$

where  $\theta' = \{\mathbf{W}', \mathbf{b}'\}$ . The weight for the decoding is set as the transpose of the encoding weight [24] in order to allow more layers to be stacked together and be fine-tuned with stochastic gradient descend (SGD).

In general, the output  $\mathbf{z}$  should be interpreted as a function of parameters  $\{\theta, \theta'\}$  as  $\mathbf{z} = g_{\theta'}(f_{\theta}(\mathbf{x}))$ . The parameters  $\{\theta, \theta'\}$  are optimized such that the reconstructed  $\mathbf{z}$  is as close as possible to the original  $\mathbf{x}$  and maximizes  $P(\mathbf{x}|\mathbf{z})$ . A typical loss function used is the mean square error (MSE), i.e.  $L(\mathbf{x}, \mathbf{z}) = \frac{1}{n}|\mathbf{x} - \mathbf{z}|^2$ .

### 3.2. Denoising Auto-encoder

The denoising auto-encoder is a variant of the basic auto-encoder. It is reported that the denoising auto-encoder can extract features more robustly than the basic auto-encoder [25]. In the denoising auto-encoder, the original data  $\mathbf{x}$  is first corrupted to  $\tilde{\mathbf{x}}$  before it is mapped to a higher representation  $f_{\theta}(\tilde{\mathbf{x}})$  by an encoder. The decoder then maps the higher representation to the output  $\mathbf{z}$  for reconstructing the original  $\mathbf{x}$ . The denoising auto-encoder is trained such that the reconstructed  $\mathbf{z}$  is as close as possible to the original data  $\mathbf{x}$ . Note that it is only during training that the denoising auto-encoder is used to reconstruct the original  $\mathbf{x}$  from the corrupted  $\tilde{\mathbf{x}}$ .

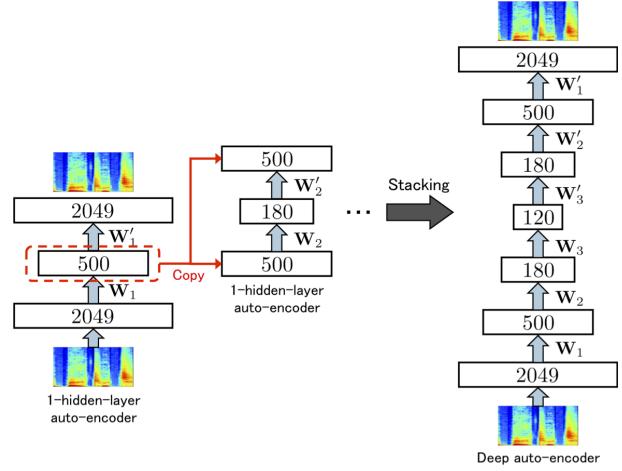
### 3.3. Deep Auto-encoder

Auto-encoder or denoising auto-encoder can be made deeper by stacking multiple layers of encoders and decoders to form a deep architecture. [26] shows that deeper architecture produces better high-level features compared to the shallow architecture up to 4 encoding and 4 decoding layers. For constructing a deep auto-encoder pre-training is widely used. In pre-training, the number of layers in a deep auto-encoder increases twice as compare to a deep neural network (DNN) when stacking each pre-trained unit. It is reported that fine-tuning with back-propagation through a deep auto-encoder is ineffective due to vanishing gradients at the lower layers [27]. To over come this issue we restrict the decoding weight as the transpose of the encoding weight following [24], that is,  $\mathbf{W}' = \mathbf{W}^T$  where  $\mathbf{W}^T$  denotes transpose of  $\mathbf{W}$ . We describe the detail of training a deep auto-encoder in the next session.

## 4. TRAINING A DEEP DENOISING AUTO-ENCODER

### 4.1. Greedy Layer-wise Pre-training

Each layer of a deep auto-encoder can be pre-trained greedily to minimize the reconstruction loss  $L(\mathbf{x}, \mathbf{z})$  of the data locally. Figure 1 shows a procedure of constructing a deep auto-encoder using



**Fig. 1.** Greedy layer-wise pre-training for constructing deep auto-encoder.

pre-training. In pre-training a 1-hidden-layer auto-encoder is trained and the encoding output of the locally trained layer is used as the input for the next layer. This layer-wise training is repeated until the desired layer size is obtained. The encoding, decoding and loss functions of each layer are represented as follows:

Layer 1 :

$$\begin{aligned} \mathbf{y}_1 &= f_{\mathbf{W}_1, \mathbf{b}_1}(\mathbf{x}), \\ \mathbf{z}_1 &= g_{\mathbf{W}'_1, \mathbf{b}'_1}(\mathbf{y}_1), \\ L(\mathbf{x}, \mathbf{z}_1) &= |\mathbf{x} - \mathbf{z}_1|^2, \\ \text{Layer } k \ (k > 1) : \\ \mathbf{y}_k &= f_{\mathbf{W}_k, \mathbf{b}_k}(\mathbf{y}_{k-1}), \\ \mathbf{z}_k &= g_{\mathbf{W}'_k, \mathbf{b}'_k}(\mathbf{y}_k), \\ L(\mathbf{y}_{k-1}, \mathbf{z}_k) &= |\mathbf{y}_{k-1} - \mathbf{z}_k|^2. \end{aligned} \quad (4)$$

Note that during the pre-training of the deep denoising auto-encoder, the input  $\mathbf{x}$ ,  $\mathbf{y}_k$  for each layer are corrupted to  $\tilde{\mathbf{x}}$  and  $\tilde{\mathbf{y}}_k$  respectively. After all layers are pre-trained, all the pre-trained layers are stacked for constructing a deep denoising auto-encoder in the same way as the deep auto-encoder.

### 4.2. Fine-tuning

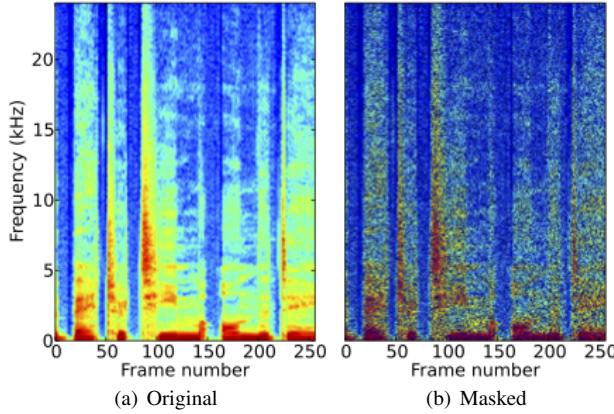
The purpose of fine-tuning is to minimize the reconstruction error  $L(\mathbf{x}, \mathbf{z})$  over the entire dataset and a model architecture using error back-propagation [28]. We use the mean square error (MSE) for the loss function of a deep auto-encoder and it is represented as follows:

$$E = \sum_{i=1}^N |\mathbf{x}^{(i)} - \mathbf{z}^{(i)}|^2, \quad (5)$$

where  $N$  is the total number of training examples. The partial derivatives w.r.t weight  $w_{i,j}^{(l)}$  is represented as follows:

$$\frac{\partial E}{\partial w_{i,j}^{(l)}} = \frac{\partial E}{\partial t_j^{(l)}} \times \frac{\partial t_j^{(l)}}{\partial w_{i,j}^{(l)}}, \quad (6)$$

where  $t_j^{(l)}$  is the fan-in input to neuron  $j$  in layer  $l$ , and  $\frac{\partial t_j^{(l)}}{\partial w_{i,j}^{(l)}} = o_i^{(l-1)}$ , where  $o_i^{(l-1)}$  is the output from neuron  $i$  at layer  $l-1$ .  $\frac{\partial E}{\partial t_j^{(l)}}$



**Fig. 2.** These figures shows parts of original and masked spectra. In the right figure black points indicated masked regions.

is the error transfer function which can be calculated recursively following

$$\frac{\partial E}{\partial t_j^{(l-1)}} = \frac{\partial o_i^{(l-1)}}{\partial t_i^{(l-1)}} \times \sum_{j=1}^L \frac{\partial E}{\partial t_j^{(l)}} \times \frac{\partial t_j^{(l)}}{\partial o_i^{(l-1)}} \quad (7)$$

where  $\frac{\partial t_j^{(l)}}{\partial o_i^{(l-1)}} = w_{i,j}^{(l)}$ . For the output tanh layer we have  $\frac{\partial o_i^{(L)}}{\partial t_i^{(L)}} = \text{sech}^2(t_i^{(L)})$ . Once we have the gradients of error function w.r.t to the weight parameters, we can fine-tune the network with error back-propagation.

#### 4.3. Corrupted data

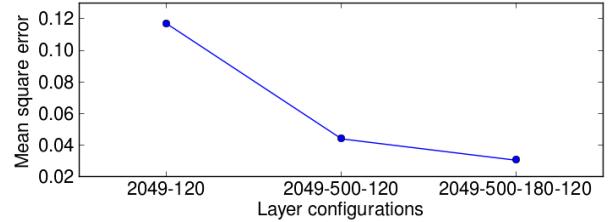
We used a masking technique reported in [25] to corrupt the training data for the denoising auto-encoder. This technique independently and randomly set the values of the training data in different dimensions to zero following a Bernoulli distribution. Figure 2 shows an example of original and masked spectra. In this figure, black points indicate masked regions.

## 5. EVALUATION

This section shows experimental results. We have evaluated the proposed auto-encoder method in the context of analysis-by-synthesis condition and text-to-speech conditions. In the text-to-speech experiments, the synthetic voices using the proposed acoustic features were modeled using two state-of-the-art speech synthesis systems: HMM and DNN.

### 5.1. Dataset

The dataset we use consists of 4569 short audio waveforms uttered by a professional English female speaker and each waveform is around 5 seconds long. For each waveform, we first extract its frequency spectra using STRAIGHT vocoder with 2049 FFT points. We then extract the low dimensional feature from each 2049-dim STRAIGHT spectrum using autoencoder. All data was sampled at 48 kHz. For comparison of the proposed method, we extracted mel-cepstral coefficients that use the same dimensions as that of auto-encoder. All other acoustic features such as log F0 and 25 aperiodicity band energies are the same for all the systems.



**Fig. 3.** Reconstruction mean square errors for auto-encoders of different architectures but same bottleneck dimension.

**Table 1.** The table lists down the hyperparameters used for training each model. lr: learning rate, m: momentum, b: batch size, s: numpy random variable weight initialization seed [29], d: masking probability of each input dimension [25].

	layer dim	lr	m	b	s	d
Deep auto-encoder	2049-500	0.001	0.9	200	8963	N.A
	500-180	0.01	0.5	50	1902	N.A
	180-120	0.01	0.9	50	6555	N.A
	Finetune	0.01	0.5	150	9781	N.A
Deep denoising auto-encoder	2049-500	0.01	0.1	150	5252	0.1
	500-180	0.01	0.5	150	7514	0.1
	180-120	0.01	0.9	100	594	0.5
	Finetune	0.001	0.9	100	2208	N.A

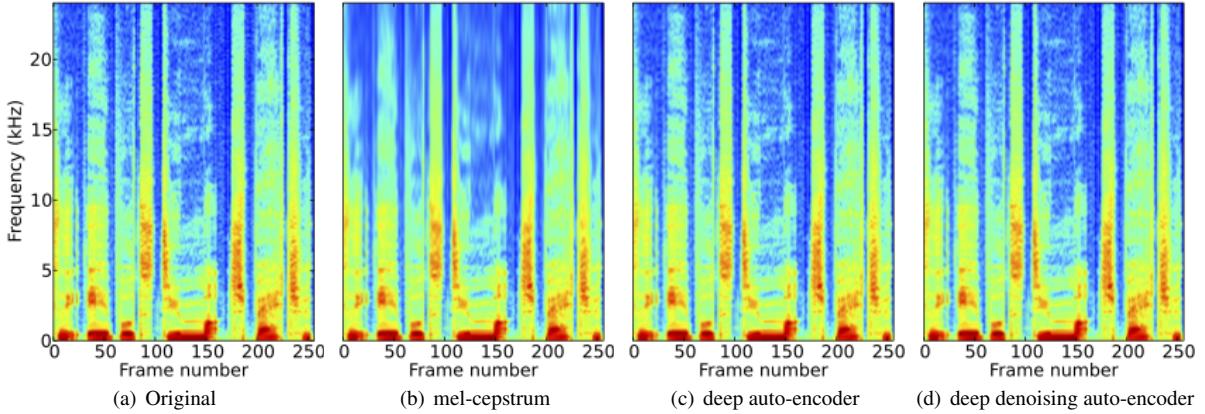
### 5.2. Configurations of the deep denoising auto-encoder

Figure 3 shows the reconstruction mean square errors of auto-encoders trained on raw frequency-warped spectrum with different number of hidden layers. It shows that the error decreases with more hidden layers, and that deep auto-encoder is better than shallow auto-encoder with the same bottleneck dimension. For the results in the rest of paper, we use architecture of the auto-encoder as 2049-500-180-120 for producing the 120-dim acoustic features, tanh units for all the layers and the inputs are 2049-dim Bark-scale-based frequency-warped spectrum, which are preprocessed with global contrast normalization. The hyperparameters used for the layer-by-layer pre-training are searched randomly and the set of values that produce the best results are selected. Table 1 shows the hyperparameters for the auto-encoders used in the experiments.

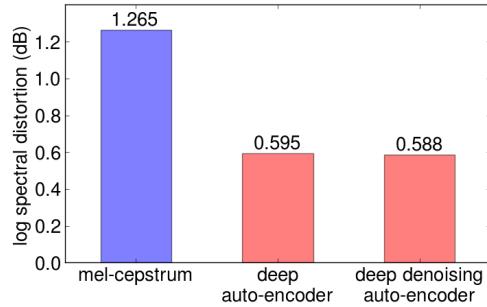
### 5.3. Analysis-by-synthesis experimental results

First we report the analysis-by-synthesis experimental results. For this evaluation, we have divided the above database into three subsets, that is, training, validation and test. The training subset was used as training data for building the auto-encoder, the validation subset was used as a stopping criteria during training to prevent overfitting, and the test subset was used for measuring log-spectral distortion and listening test.

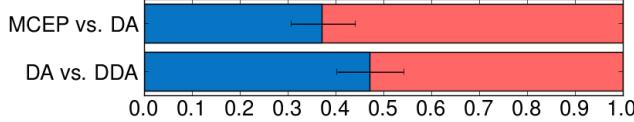
Figure 4 shows the original and reconstructed spectra using each technique (mel-cepstral analysis, deep auto-encoder, deep denoising auto-encoder). We can clearly see that the deep auto-encoders reconstruct high-frequency parts more precisely than mel-cepstral analysis. Figure 5 shows log spectral distortion between the original spectra and reconstructed spectra, calculated on the test subset. We can observe that the deep auto-encoder has reduced the distortion significantly compared to the mel-cepstral analysis and denoising version further reduced the distortion. Figure 6 shows subjective preference scores of these methods. The number of listeners



**Fig. 4.** Original and reconstructed spectra using each technique.



**Fig. 5.** log spectral distortion between the original and reconstructed spectra.

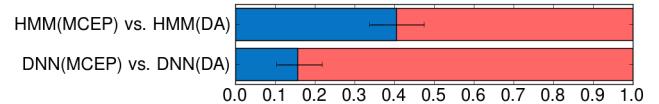


**Fig. 6.** Results of preference tests using analysis-by-synthesis speech samples. In this figure, MCEP, DA and DDA refer to mel-cepstral analysis, deep auto-encoder and deep denoising auto-encoder respectively.

that performed this test were seven. They have participated in two preference tests. In the first preference test, they were asked to compare deep auto-encoder (DA) with mel-cepstral analysis (MCEP). In the second preference test, they were asked to compare deep auto-encoder with deep denoising autoencoder (DDA). From the figure, we can see that deep auto-encoder based speech samples sound more natural than mel-cepstral analysis based speech samples. Deep denoising auto-encoder reduced the distortion, however, perceptual difference between clean and denoising auto-encoder is not statistically significant.

#### 5.4. Text-to-speech experimental results

Next we report the text-to-speech experimental results. For the HMM-based speech synthesis, we have used a hidden semi-Markov model and the observation vectors for the spectral and excitation parameters contained static, delta and delta-delta values, with one stream for the spectrum, three streams for F0 and one for the band-limited aperiodicity. The context-dependnet labels are built using the



**Fig. 7.** Results of preference tests using text-to-speech samples. In this figure, MCEP and DA refer to mel-cepstral analysis and deep auto-encoder for the acoustic feature extraction, and HMM and DNN are the acoustic models.

pronunciation lexicon Combilex [30]. For the DNN-based speech synthesis, we have trained a five-hidden-layer DNN for mapping between linguistic contexts and auto-encoder-based or mel-cepstral acoustic features. The number of units in each of the hidden layers was set to 512. Random initialisation was used in a similar way to [8]. Figure 7 shows subjective preference scores where we have compared the proposed auto-encoder feature with the conventional mel-cepstral feature in each of the HMM-based speech synthesis and the DNN-based speech synthesis systems. Listeners are the same as those for Figure 6. We can see that synthetic speech using the proposed feature sound more natural than the conventional mel-cepstral features in both the synthesis methods. The proposed feature seems to suit the DNN-based speech synthesis better, but, this requires further investigation.

## 6. CONCLUSIONS

In this paper we have proposed the deep denoising auto-encoder technique to extract better acoustic features for speech synthesis. We have compared the new stochastic feature extractor with the conventional mel-cepstral analysis in the analysis-by-synthesis and text-to-speech experiments and have confirmed that the proposed method can increase the quality of synthetic speech in both the conditions.

Our future work includes the improvement of the deep denoising auto-encoder. In this paper, we have used the simplest noise, i.e. masking and the improvement was observed only from objective evaluation. We shall use or design different types of noises to improve the deep denoising auto-encoder for speech synthesis further.

## 7. REFERENCES

- [1] H. Zen, K. Tokuda, and A. W. Black, “Statistical parametric speech synthesis,” *Speech Communication*, vol. 51, pp. 1039–1064, 2009.
- [2] T. Yoshimura, K. Tokuda, T. Masuko, T. Kobayashi, and T. Kitamura, “Speaker interpolation in HMM-based speech synthesis system,” *Proceedings of Eurospeech 1997*, pp. 2523–2526, 1997.
- [3] T. Yoshimura, K. Tokuda, T. Masuko, T. Kobayashi, and T. Kitamura, “Simultaneous modeling of spectrum, pitch and duration in HMM-based speech synthesis,” *Proceedings of Eurospeech 1999*, pp. 2347–2350, 1999.
- [4] R. Tsuzuki, H. Zen, K. Tokuda, T. Kitamura, M. Bulut, and S. Narayanan, “Constructing emotional speech synthesizers with limited speech database,” *Proceedings of ICSLP*, vol. 2, pp. 1185–1188, 2004.
- [5] J. Yamagishi, K. Onishi, T. Masuko, and T. Kobayashi, “Acoustic modeling of speaking styles and emotional expressions in HMM-based speech synthesis,” *IEICE Transactions on Information & Systems*, vol. E88-D, no. 3, pp. 502–509, 2005.
- [6] L.-H. Chen, T. Raitio, C. Valentini-Botinhao, J. Yamagishi, and Z.-H. Ling, “DNN-based stochastic postfilter for HMM-based speech synthesis,” *Proceedings of Interspeech*, pp. 1954–1958, 2014.
- [7] H. Kawahara, I. Masuda-Katsuse, and A. Chevaigne, “Restructuring speech representations using a pitch-adaptive time-frequency smoothing and an instantaneous-frequency-based F0 extraction: Possible role of a repetitive structure in sounds,” *Speech Communication*, vol. 27, pp. 187–207, 1999.
- [8] H. Zen, A. Senior, and M. Schuster, “Statistical parametric speech synthesis using deep neural networks,” *Proceedings of ICASSP*, pp. 7962–7966, 2013.
- [9] Z.-H. Ling, L. Deng, and D. Yu, “Modeling spectral envelopes using restricted Boltzmann machines and deep belief networks for statistical parametric speech synthesis,” *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 21, pp. 2129–2139, 2013.
- [10] Y. Fan, Y. Qian, F. Xie, and F. K. Soong, “TTS synthesis with bidirectional LSTM based recurrent neural networks,” *Proceedings of Interspeech*, pp. 1964–1968, 2014.
- [11] R. Fernandez, A. Rendel, B. Ramabhadran, and R. Hoory, “Prosody contour prediction with long short-term memory, bidirectional, deep recurrent neural networks,” *Proceedings of Interspeech*, pp. 2268–2272, 2014.
- [12] T. N. Sainath, B. Kingsbury, and B. Ramabhadran, “Autoencoder bottleneck features using deep belief networks,” *Proceedings of ICASSP*, pp. 4153–4156, 2012.
- [13] J. Gehring, Y. Miao, F. Metze, and A. Waibel, “Extracting deep bottleneck features using stacked auto-encoders,” *Proceedings of ICASSP*, pp. 3377–3381, 2013.
- [14] A. L. Maas, Q. V. Le, T. M. O’Neil, O. Vinyals, P. Nguyen, Andrew Ng, and Y., “Recurrent neural networks for noise reduction in robust ASR,” *Proceedings of Interspeech*, pp. 22–25, 2012.
- [15] T. Ishii, H. Komiyama, T. Shinozaki, Y. Horiuchi, and S. Kuroiwa, “Reverberant speech recognition based on denoising autoencoder,” *Proceedings of Interspeech*, pp. 3512–3516, 2013.
- [16] X. Feng, Y. Zhang, and J. Glass, “Speech feature denoising and dereverberation via deep autoencoders for noisy reverberant speech recognition,” *Proceedings of ICASSP*, pp. 1778–1782, 2014.
- [17] L. Deng, M. Seltzer1, D. Yu, A. Acero, A. Mohamed, and G. Hinton, “Binary coding of speech spectrograms using a deep auto-encoder,” *Proceedings of Interspeech*, pp. 1692–1695, 2010.
- [18] X. Lu, Y. Tsao, S. Matsuda1, and C. Hori, “Speech enhancement based on deep denoising autoencoder,” *Proceedings of Interspeech*, pp. 436–440, 2013.
- [19] N. Kumar and A. G. Andreou, “Heteroscedastic discriminant analysis and reduced rank hmms for improved speech recognition,” *Speech Communication*, pp. 283–297, 1998.
- [20] M. J. F. Gales, “Maximum likelihood multiple subspace projections for hidden Markov models,” *Speech and Audio Processing, IEEE Transactions on*, vol. 10.
- [21] S. J. D. Prince and J. H. Elder, “Probabilistic linear discriminant analysis for inferences about identity,” *ICCV*, pp. 1–8, 2007.
- [22] P. Kenny, “Bayesian speaker verification with heavy-tailed priors,” *Odyssey*, p. 14, 2010.
- [23] L. Lu and S. Renals, “Probabilistic linear discriminant analysis for acoustic modelling,” *Signal Processing Letters, IEEE*, pp. 702–706, 2014.
- [24] G. E. Hinton and R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” *Science* 28, vol. 313, no. 5786, pp. 504–507, 2006.
- [25] P. Vincent, H. Larochelle, Y. Bengio, and P. Manzagol, “Extracting and composing robust features with denoising autoencoders,” *ICML*, pp. 1096–1103, 2008.
- [26] D. Yu and M. Seltzer, “Improved bottleneck features using pretrained deep neural networks,” *Proceedings of Interspeech*, pp. 237–240, 2011.
- [27] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber, “Gradient flow in recurrent nets: the difficulty of learning long-term dependencies,” *CiteSeer*, 2001.
- [28] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1,” pp. 318–362, 1986.
- [29] I. Sutskever, J. Martens, George E. Dahl, and Geoffrey E. Hinton, “On the importance of initialization and momentum in deep learning,” *ICML*, pp. 1139–1147, 2013.
- [30] K. Richmond, R. Clark, and S. Fitt, “On generating combilex pronunciations via morphological analysis,” *Proceedings of Interspeech*, pp. 1974?–1977, 2010.
- [31] J. Bergstra and Y. Bengio, “Random search for hyperparameter optimization,” *The Journal of Machine Learning Research*, vol. 13, pp. 281–305, 2012.
- [32] X. Feng, Y. Zhang, and J. Glass, “Speech feature denoising and dereverberation via deep autoencoders for noisy reverberant speech recognition,” *Proceedings of ICASSP*, pp. 1759–1763, May 2014.